Concept WebDAV

Your Rating: Results: 93 rates

Implemented in 4.2

See WebDAVmodule documentation.

Official Documentation Available

This topic is now covered in WebDAV module.

Introduction

WebDAV protocol allows clients to map resources of compliant server as remote directories, thus greatly simplifying resource update and manipulation.

Evaluate existing implementations, choose the best base for Magnolia WebDAV, estimate effort for full implementation.

Goals

- provide mechanism to access content of various workspaces of Magnolia using dav protocol.

WebDAV information sources

- general info
- specifications

- RFC 2518 (WebDAV - HTTP Extensions for Distributed Authoring)
- RFC 3253 (DeltaV - Versioning Extensions to WebDAV)
- RFC 3648 (Ordered Collections Protocol)
- RFC 3744 (Access Control Protocol)
- draft-reschke-webdav-search (WebDAV SEARCH; previously DASL - DAV Searching and Locating)
- draft-ietf-webdav-bind (Binding Extensions to Web Distributed Authoring and Versioning (WebDAV))

Existing WebDAV server implementations

The list below is not complete, only includes implementations considered as possible base for implementing support of webdav in Magnolia.
All implementations below are done as servlets.

- Tomcat webdav support
- Randombits/Confluence webdav plugin
- JackRabbit webdav library

Current Status

Nodes mapping

- `mgnl:folder` mapped as folders
- `mgnl:content` mapped as files
- For layout and templates workspaces `text` property value is mapped as the content of the file

magnolia view (layout workspace)	jcr	filesystem	FS view
css	mgnl:folder	folder	css
.basic	.mgnl:content	.file	basic.css
..text	..jcr:property	..content of the file	..

Since we map only one property into the content of the file, rest of the properties are hidden to the filesystem and depend on the node they belong to. This means that deletion of the file, results in deletion of the node and all its properties and sub nodes (MetaData). Creation of file in filesystem results in creation appropriate node in Mangolia workspace (incl. MetaData subnode with some default values).

Problems

Most problems at the moment evolve around the way different programs handle file updates. There are basically 3 different scenarios of how different programs updates files:

1. vi style - move original `file.css` to `file.css~` and save updated content in new `file.css`
2. edit style - delete original `file.css` and save updated content to new `file.css`
3. eclipse style - just overwrite content of the file without deleting it first

The first two scenarios are most common for simple editors and lead to loss of metadata due to deletion of the file.
The third scenario seems to be preferred one by more complex applications (eclipse, cssedit, scream) and is fine as it preserves the metadata.

While we can, as suggested by some, simply ignore this and state that this is a problem of the editor and the way it handles files, it will cause problems in Magnolia. Also since it is unpredictable I personally do not think we can simply ignore it. If we do, it might lead to situations where parts of the site will not be renderable due to missing `MetaData` (e.g. template information) and will require manual corrections.

Of course the situation is similar in case of new content added via webdav, but in this case it is expected that `MetaData` will not be set.

At this moment I can see following possible solutions:

1. do not allow deletion and moving of the files via webdav - in this case any editors attempting to do so will fail early and will not cause any damage to magnolia data.
    - the caveat is that it basically renders useless half of the features for which webdav would be preferred access option
2. use configurable set of default `MetaData`(in case of layout workspace, only critical is the template name) - in this case we can set those even for new content
    - the caveat here is that `MetaData` for edited content will in some cases change without an obvious (to the customer) reason for example in case when they had previously set different template for the content then the one configured in the default settings for `MetaData`
3. change the exported structure of the nodes in repositories we want to expose - the node with `MetaData` should be exposed as folder and the property with data ("text" in case of layout workspace) would be exposed as file in that directory. This way deleting the directory deletes content from the workspace, any changes to the file representing `text`property affect only value of that property (Deleting the file either removes the property or sets it to empty string).
    - the caveat I see here is that the structure in filesystem will be more complicated to use - having one file per directory might seem cumbersome to some people

Tasks

Task List

☑ evaluate use of Tomcat and RandomBits server impls
☑ evaluate and estimated effort necessary for full standard implementation
☑ evaluate JR webdav support
☐ **- have webdav access working from browser (read-only).**
☐ **- have webdav access working from other clients (issues with authentication).**
☐ **- solve write issues when accessing mounted share share on linux**
☐ **- test access from mac.**
☐ **- test access from windows.**
☐ **- provide custom mapping for nodes to expose only one property of the node under the name of the node itself and map only folders as folders.**
☐ **- ensure session handling is compatible with rest of Magnolia - choose proper RepositoryStrategy (have to be changed at the Servlet level).**

Notes:

- according to http://httpd.apache.org/docs/2.0/mod/mod_dav.html most clients support `digest` authentication and basic over ssl.
- Common WebDAV issues:
  - client specific problems:
  - path encoding differs for mac and windows clients
  - different sets of acceptable characters

`magnolia-module-webdav` is now committed in svn. Editing of the `web.xml` is no longer necessary, just build and deploy the module to test it.
To mount the website workspace as webdav share you have to run something along the:

```
mount -t davfs \
http://localhost:8080/magnoliaAuthor/.webdav/layout \
/media/locdav
```

Implementation details of considered solutions

Tomcat webdav library

The tomcat webdav support is tight to providing html authoring via webdav and implementation is all based about this fact. There is no clear separation of concerns which would make it difficult trying to reuse this implementation to serve magnolia content.

Randombits/Confluence webdav plugin

The Randombits webdav plugin provides nice and clean separation of webdav related binding and request processing and backend functionality (serving content from confluence). By exchanging backend and implementing magnolia resource (as opposed to confluence resource) we should be able to reuse this plugin. Randombits webdav plugin have been contributed to confluence. Plugin is using bsd like license which might be in conflict in with our licensing schema.

Clean full implementation of WebDAV protocol and associated specifications

Other option it to implement support from the beginning. To provide full support would be an extensive amount of work (first guess 4-5 man weeks). We can limit time necessary by providing only partial support, yet it might be limiting to certain clients only (each client has little bit specific way of using different features of the server to create and maintain connection and serve the content).

JR WebDAV library

JackRabbit library provide very extensive support - implements all the specs listed above. Limitation here seems to be necessity of using JR authentication. However it is possible to specify default authentication against repository to be used when no other credentials are supplied. Since Magnolia authentication is checked in the logon filter, it is possible to protect access to webdav servlet via existing Magnolia filter chain and only bypasss c `ms` part of the chain. Since specification mandates the dav resources are not cached, we should at least consider bypassing the cache filter as well.
One existing problem with using Magnolia filter chain for handling authentication to webdav access is the fact that no client (except browsers) is able to cope with the FormLoginCallback. The specification mandates for clients to support Basic authentication only (implemented in `BasicLoginCallback` in Magnolia}. Therefore if used we need to extend current callback resolving mechanism to be path aware and allow different callback to be used based on the URI of the request.
The biggest advantage over all other approaches seems to be effort necessary for implementation. JR WebDAV library provides abstract servlet implementation which requires from extending class to provide only `public Repository getRepository();` method implementations. So all effort necessary to provide full webdav support is implementation of the method in custom servlet, configuration of filter chain and configuration of security.